Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра автоматизации обработки информации (АОИ)

ПРОЕКТИРОВАНИЕ ЧЕЛОВЕКО-МАШИННОГО ИНТЕРФЕЙСА

Методические указания к лабораторным работам,
организации самостоятельной работы и выполнению курсовой работы
для студентов направления
«Программная инженерия»
(уровень бакалавриата)
(заочная форма обучения)

2018

Петкун Татьяна Александровна

Проектирование человеко-машинного интерфейса: Методические указания к лабораторным работам, организации самостоятельной работы и выполнению курсовой работы для студентов направления «Программная инженерия» (уровень бакалавриата) (заочная форма обучения) / Т.А. Петкун. – Томск, 2018. – 34 с.

Оглавление

1 Введение

Проектирование человеко-машинного интерфейса – дисциплина, имеющая дело с разработкой, развитием и применением интерактивных компьютерных систем с точки зрения требований пользователя, а также с изучением явлений их окружающих. Этот курс предназначен для программистов и пользователей и обеспечивает изучение компьютерных технологий с акцентом на разработку и развитие пользовательского интерфейса.

Проектирование человеко-машинного интерфейса – это дисциплина, объединяющая знания из нескольких областей: психологии познания, проектирования программного обеспечения и компьютерных систем, социологии и организации бизнеса, эргономики и системного анализа, управления процессами и промышленного дизайна. Внедрение компьютеров практически во все стороны жизни требует от современного специалиста в области компьютерных технологий умения разработать или адаптировать пользовательский интерфейс под широкий класс пользователей, обеспечить эффективное использование компьютерных систем в разных приложениях.

2 Методические указания к проведению лабораторных работ

2.1 Лабораторная работа «Создание графического интерфейса с использованием средств визуального программирования»

Цель работы

Приобретение практических навыков создания пользовательского интерфейса. Знакомство с технологией визуального программирования.

Ход работы

Под руководством преподавателя в любой доступной среде программирования создать интерфейсы к пяти следующим заданиям.

Задание 1.

Создайте проект, содержащий 5 кнопок TButton и компонент Timage (со свойством Visible) со следующими функциями:

– первая кнопка прячет/показывает рисунок;

– вторая кнопка сдвигает первую на 10 пикселей вверх (когда кнопка достигнет верхней границы формы, вернуть кнопку на исходную позицию; учесть реальные ограничения, связанные с размером формы);

– третья кнопка включает/выключает системные кнопки главного окна (посмотрите свойство BorderStyle);

– четвертая кнопка плавно делает форму прозрачной и возвращает первоначальный вид формы назад (используем свойства формы AlphaBlendValue, AlphaBlend и процедуру Sleep());

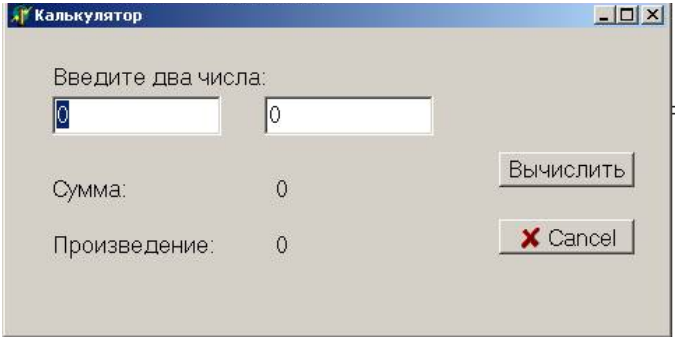
— пятая кнопка закрывает форму.

Задание 2. Калькулятор

Создать проект, который реализует работу простейшего калькулятора. При вводе чисел предусмотреть проверку ошибок ввода, например, с использованием следующего фрагмента программы:

```
Try
  a:= StrToInt(Edit1.Text);   {Попытка перевести строку в число}
Except
   ShowMessage('Ошибка ввода');  {Если не получилось, выдаем сообщение об ошибке}
   Edit1.SetFocus; {Устанавливаем фокус на компонент, в котором произошла ошибка ввода}
```

```
  Exit;  {Прекращаем работу процедуры}
End;
```



Внешний вид приложения

В проекте участвуют объекты:

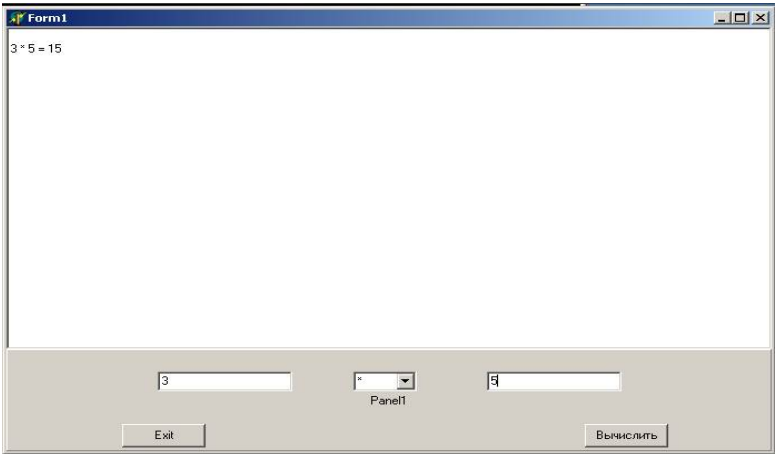
- форма - TForm,
- кнопка - TButton со свойством Caption с методом – OnClick,
- кнопка -TBitBtn со свойством Kind,
- две строки ввода TEdit со свойством Text,
- пять текстовых меток Tlabel со свойством Caption.

Задание 3.

Создать проект, имитирующий работу микрокалькулятора. Программа позволяет вводить два операнда и в выпадающем списке выбрать знак математического действия, после чего в многострочном редакторе отображается результат.

В проекте участвуют объекты:

- форма – Tform, со свойством Caption,
- кнопки - TButton со свойством Caption, с методом – OnClick,
- две строки ввода TEdit со свойством Text,
- метки Tlabel со свойством Caption,
- многострочный редактор TMemo со свойством Lines;
- компонент ComboBox.

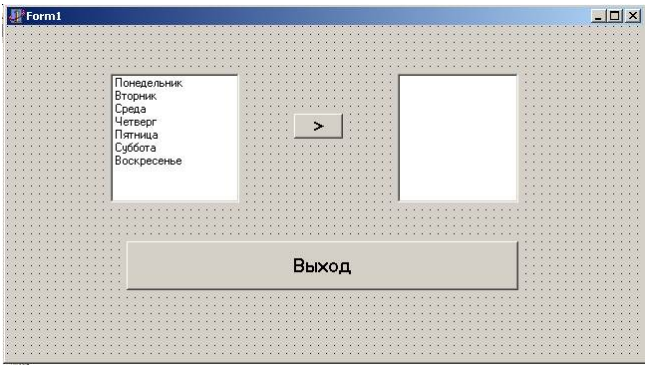


Внешний вид приложения

Примечание: знак математической операции выводим с использованием компонента ComboBox (страница Standard). Установите для него Width=41. Раскройте диалоговое окно свойства Items этого компонента и введите четыре строки со знаками математических действий. Свойство ItemIndex содержит индекс сфокусированного элемента. Если ItemIndex=–1, то ни одна из строк компонента не выбрана. Items – набор строк, показываемых в компоненте. Добавить строку S в многострочный редактор можно так: Memo1.Lines.Add(S).

Задание 4.

Создайте проект, позволяющий выбирать из левого списка элемент и если нажата кнопка ‘ > ‘, то этот элемент появляется в списке справа. Не добавлять в правый список строку, если такая уже есть в этом списке!



В проекте участвуют объекты:

- форма – Tform, со свойством Caption,
- кнопки - TButton со свойством Caption, с методом – OnClick,
- два компонента TlistBox со свойствами Items, ItemIndex.

Задание 5.

Модальной называется форма, которая должна быть закрыта перед обращением к любой другой форме данного приложения. Если пользователь пытается перейти в другую форму, не закрыв текущую модальную форму, то Windows блокирует эту попытку. Модальные формы часто называют ***диалогами*** . Диалоговые формы обычно используются при выполнении таких операций, как ввод данных, открытие или сохранение файлов, вывод информации о приложении, установка параметров приложения и т.п.

Для ***отображения*** формы в модальном режиме служит метод ShowModal:

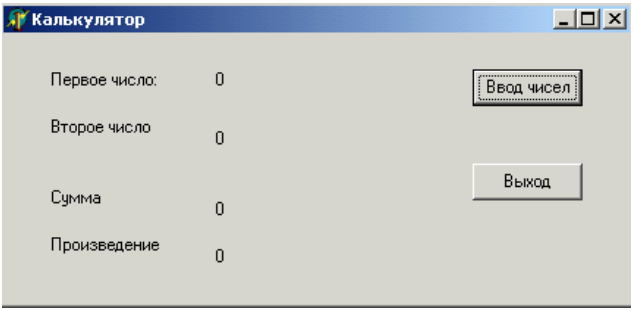
```
Procedure TForm1.Bottun1Click (Sender: TObject);
Begin
   Form2.ShowModal;
End;
```

При закрытии модальной формы функция ShowModal возвращает значение свойства ModalResult, Возможные значения этого свойства:

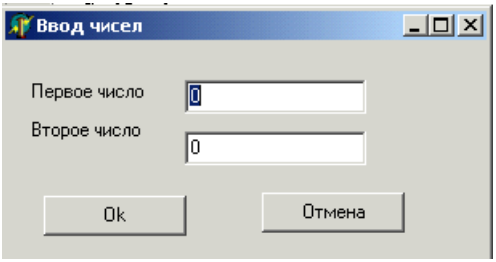
mrNone, mrOk, mrCancel, mrYes, mrNo и другие.

Многие формы можно отображать и в немодальном режиме. Метод Show открывает немодальное окно.

Создайте главную форму и вспомогательную (модальную) форму для задачи «Калькулятор».



Главная форма приложения



Вспомогательная форма

В проекте участвуют объекты:

- формы – Tform1, Tform2, со свойством Caption, с методом(Tform2) – ShowModal,
- кнопки - TButton со свойством Caption, с методом – OnClick,
- строки ввода – Tedit,
- Метки TLabel.

2.2 Лабораторная работа «Панели инструментов, меню, использование таблиц»

Цель работы

Научиться использовать соответствующие компоненты для работы с таблицами, проектировать интерфейс, который позволяет пользователю использовать различные способы для работы с программой.

Теоретические основы

Для визуализации работы с двумерным массивом будем использовать компонент TStringGrid со страницы Additional палитры компонентов, предназначенный для создания таблиц, в ячейках которых располагаются произвольные текстовые строки

Таблица делится на две части – фиксированную и рабочую. Фиксированная служит для показа заголовков столбцов/рядов и для ручного управления их размерами. Обычно фиксированная часть занимает крайний левый столбец и самый верхний ряд таблицы, однако с помощью свойств FixedCols и FixedRows можно задать другое количество фиксированных столбцов и рядов (если эти свойства имеют значение 0, таблица не содержит фиксированной зоны). Рабочая часть – это остальная часть таблицы. Она может содержать произвольное количество столбцов и рядов, более того, эти величины могут изменяться программно. Рабочая часть может не умещаться целиком в пределах окна

компонента, в этом случае в него автоматически помещаются нужные полосы прокрутки. При прокрутке рабочей области фиксированная область не исчезает, но меняется ее содержимое – заголовки строк и рядов.

Свойство	Тип	Комментарий
Cells[ACol,ARow: Iteger]	String	Определяет содержимое ячейки с табличными координатами (ACol, ARow)
Cols[ACol: Integer]	TStrings	Все строки с номером ACol
Rows[ARow: Integer]	TStrings	Все колонки с номером ARow
ColCount	Integer	Общее количество столбцов
RowCount	Integer	Общее количество строк
FixedCols	Integer	Количество фиксированных (заголовочных) столбцов
FixedRow s	Integer	Количество фиксированных (заголовочных) строк
Col	Integer	Номер столбца текущей ячейки
Row	Integer	Номер ряда текущей ячейки
	TGridOptions	Данное свойство множественного типа определяет ряд дополнительных параметров таблицы. В частности, нам наиболее важны следующие:
	goEditing	Разрешается редактирование ячеек таблицы
	goTabs	Разрешается перемещение по ячейкам с помощью клавиши Tab и комбинации Shift+Tab

Центральным свойством компонента является Cells – двумерный массив ячеек, каждая из которых может содержать произвольный текст. Конкретная ячейка определяется парой чисел – номером столбца и номером ряда, на пересечении которых она находится (нумерация начинается с нуля). Свойство Cells имеет тип String, поэтому программа может легко прочитать или записать содержимое нужной ячейки. Например:

```
Cells[1,1] := 'Верхняя ячейка рабочей зоны';
```

Количество ячеек по каждому измерению хранит пара свойств ColCount (количество столбцов) и RowCount (количество строк).

Значения этих свойств, и, следовательно, размеры таблицы могут меняться как на этапе разработки программы, так и в ходе ее работы, однако их значения должны быть как минимум на единицу больше соответственно значений в свойствах FixedCols и FixedRows, определяющих размеры фиксированной зоны. Содержимое ячеек можно редактировать.

В данной работе наше приложение должно быть красиво оформлено в виде пиктограмм. Пиктограммы будут назначаться пунктам меню и кнопкам панели инструментов. Для того, чтобы можно было назначить картинку любому из этих элементов, необходимо вначале установить в свойстве ImageList содержащего его компонента используемый список картинок, а затем в свойстве ImageIndex каждого элемента указать номер картинки из списка. Однако если наша программа основана на концепции действий (Action), вместо прямого задания значения свойства ImageIndex пункта меню или кнопки панели инструментов следует указать значение свойства ImageIndex соответствующего компонента TAction.

Для определения действия необходимо на форму поместить компонент типа TActionList, который может содержать в себе множество действий. Затем нужно дважды щелкнуть на нем мышкой, при этом откроется редактор действий. Нажатием на клавишу Insert можно добавлять новые действия. При выборе мышкой действия в списке, оно становится доступным в инспекторе объектов. Список основных свойств приведен ниже в таблицах 1 и 2.

Таблица 1.Основные свойства объектов типа TAction

Свойство	**Тип**	**Комментарий**
Caption	String	Название действия в меню
Category	String	Категория – используется для упорядочивания действий внутри TActionList
Checked	Boolean	Отмечены ли галочкой пункты меню и нажаты ли соответствующие кнопки
Enabled	Boolean	Разрешена ли команда
Hint	String	Всплывающая подсказка для кнопок
ImageIndex	Integer	Код картинки в связанном

		списке картинок
ShortCut	TshortCut	Код горячей клавиши для вызова действия
Visible	Boolean	Видимы ли пункты меню и кнопки

Таблица 2. Основные события объектов типа TAction

Свойство	Комментарий
OnExecute	Выполнение действия
OnUpdate	Обновление информации о действии. Здесь можно изменить любые свойства действия в зависимости от текущего состояния программы. Обычно с помощью этого события запрещают недоступные команды.

После того, как созданы все необходимые действия, их необходимо назначить соответствующим пунктам меню и кнопкам. У этих компонентов имеется свойство Action, значение которого можно установить в инспекторе объектов с помощью выпадающего списка всех доступных в форме действий.

В нашем приложении необходимо сделать назначение действий для главного меню, локального меню и кнопкам панели.

При создании элементов меню, соответствующих действиям, не нужно указывать заголовки. Нужно задать свойство Action элемента меню, при этом заголовок, код картинки, горячая клавиша и обработчик события будут автоматически взяты из компонента TAction .

Замечание *Не забудьте указать свойство ImageList компонетов меню и панели инструментов из выпадающего списка в инспекторе объектов для отображения в них картинок.*

Локальное меню обычно появляется на экране при нажатии правой кнопки мыши на визуальных компонентах. Для этого у этих компонентов должно быть установлено свойство PopupMenu в инспекторе объектов. В нашем случае достаточно назначить свойство PopupMenu только для формы. Тогда нажатие правой кнопки мышки в любом месте формы будет вызывать локальное меню.

Для создания панели инструментов с кнопками используем компонент Ttoolbar. Для добавления в него новых кнопок необходимо нажать правую кнопку мыши для вызова локального меню и выбрать соответствующий пункт. Как и пункты меню, кнопки имеют свойство Action, при установке значения которого из выпадающего списка кнопки автоматически получат все необходимые свойства (текст надписи, всплывающую подсказку, код картинки, обработчик нажатия). По умолчанию кнопки на панели инструментов отображаются только с картинкой без сопроводительной надписи. Для отображения с надписью необходимо указать свойство ShowCaption равным True.

В большинстве современных приложений панели инструментов можно свободно перетаскивать по экрану с помощью мышки.

Для того, чтобы это можно было делать, в Delphi имеется компонент типа TControlBar, который дополняет все помещаемые на него другие компоненты рамочкой и двумя вертикальными полосками слева для перетаскивания. Обычно в TControlBar помещают панели инструментов TToolBar. Поэтому перед размещением панелей инструментов на форме сначала необходимо поместить компонент TControlBar, а в него затем – панели инструментов.

Задание

Создать проект, реализующий работу с двумерным массивом. Приложение должно быть снабжено главным меню, локальным меню и панелью инструментов.

Таблица 3. Список действий, создаваемых в приложении

Действие	Описание
ActionInput	Заполнить матрицу с помощью датчика случайных чисел
ActionClear	Очистить матрицу. Вернуть первоначальную размерность
ActionMatr	Смотрите задание
ActionExit	Выход из программы
ActionAbout	Выдает краткую информацию об авторе

Таблица 4. Варианты задания

Номер задания	Описание
1	Удалить строку, содержащую максимальный элемент.

2	Удалить столбец, содержащий максимальный элемент.
3	Отсортировать матрицу по возрастанию элементов первого столбца.
4	Отсортировать матрицу по убыванию элементов первой строки.
5	Удалить строку и столбец, на пересечении которых находится минимальный элемент матрицы.

2.4 Лабораторная работа «Создание MDI-приложения»

Цель работы

Познакомиться с технологией создания многодокументного интерфейса

Теоретические основы

Термин MDI (Multiple Document Interface) дословно означает многодокументный интерфейс и описывает приложения, способные загрузить и использовать одновременно несколько документов или объектов. Примером такого приложения может служить диспетчер файлов (FileManager).

Обычно MDI-приложения состоят минимум из двух форм – родительской и дочерней. Свойство родительской формы FormStyle установлено равным fsMDIForm. Для дочерней формы установите стиль fsMDIChild.

Родительская форма служит контейнером, содержащим дочерние формы, которые заключены в клиентскую область и могут перемещаться, изменять размеры, минимизироваться или максимизироваться. В вашем приложении могут быть дочерние формы разных типов, например одна – для обработки изображений, а другая – для работы с текстом.

Создание форм

В MDI-приложении, как правило, требуется выводить несколько экземпляров классов формы. Поскольку каждая форма представляет собой объект, она должна быть создана перед использованием и освобождена, когда в ней больше не нуждаются.

Автоматическое создание форм

По умолчанию при запуске приложения Delphi автоматически создает по одному экземпляру каждого класса форм в проекте и освобождает их при завершении программы. Автоматическое создание

обрабатывается генерируемым Delphi кодом в трех местах.

Первое – раздел интерфейса в файле модуля формы.

```
type
 TForm1 = class (TForm)
 private
    {Закрытые объявления.}
 public
    {Открытые объявления.}
 end;
```

В данном фрагменте кода объявляется класс TForm1. Вторым является место, в котором описывается переменная класса.

```
var Form1: TForm1;
```

Здесь описана переменная Form1, указывающая на экземпляр класса TForm1 и доступная из любого модуля. Обычно она используется во время работы программы для управления формой. Третье место находится в исходном тексте проекта, доступ к которому можно получить с помощью меню View/ Project Source. Этот код выглядит как:

```
Application.CreateForm(TForm1, Form1);
```

Процесс удаления форм обрабатывается с помощью концепции владельцев объектов: когда объект уничтожается, автоматически уничтожаются все объекты, которыми он владеет. Созданная описанным образом форма принадлежит объекту Application и уничтожается при закрытии приложения.

Динамическое создание форм

Хотя автоматическое создание форм полезно при разработке SDI-приложений, при создании MDI-приложении оно, как правило, неприемлемо.

Для создания нового экземпляра формы используйте конструктор Create класса формы. Приведенный ниже код создает новый экземпляр TForm1 во время работы программы и устанавливает его свойство Caption равным 'New Form'.

```
Form1:= TForm1.Create(Application);
Form1.Caption:= 'New Form';
```

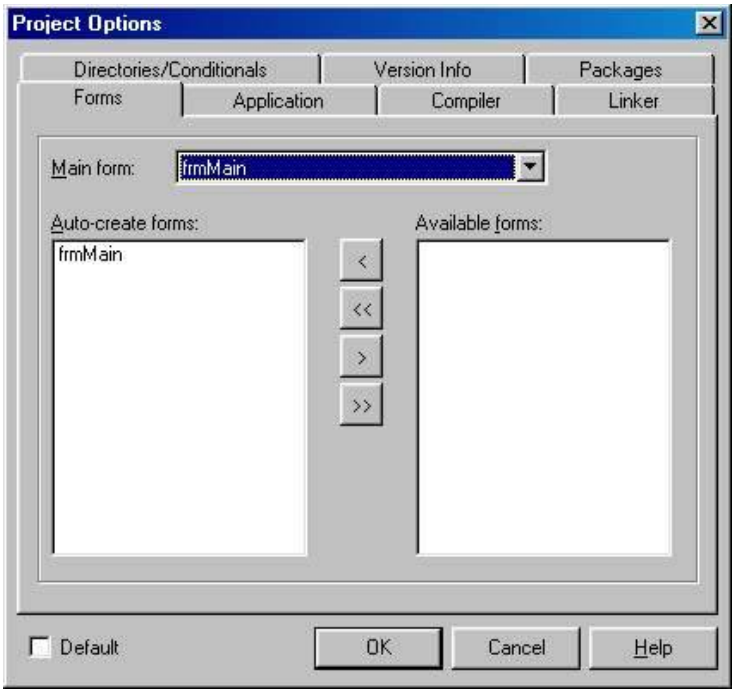
В приведенном ниже коде Form1 указывает только на последнюю созданную форму. Если вам это не нравится, воспользуйтесь

приведенным ниже кодом – возможно, он более точно отвечает вашим запросам:

```
with TFormI.Create(Application) do          Caption:= 'New Form';
```

Замечание: При разработке MDI-приложения метод Show не нужен, так как Delphi автоматически показывает все вновь созданные дочерние MDI-формы. В случае SDI-приложения вы обязаны использовать метод Show.

Даже при динамическом создании форм Delphi попытается навязать вам свои услуги по созданию экземпляра каждой формы. Чтобы отказаться от них, воспользуйтесь диалоговым окном Project Options, изображенным на рисунке, и удалите классы форм из списка Auto-create forms.



Диалоговое окно Project Options

Если вы захотите получить доступ к отдельному дочернему экземпляру класса, используйте свойство MDIChildren.

MDI-свойства TForm

Объект TForm имеет несколько свойств, специфичных для MDI-приложений.

ActiveMDIChild

Это свойство возвращает дочерний объект TForm, имеющий в текущее время фокус ввода. Оно полезно, когда родительская форма содержит панель инструментов или меню, команды которых распространяются на открытую дочернюю форму.

Например, представим, что проект использует дочернюю форму, содержащую элемент TMemo, названный memDailyNotes. Имя класса этой дочерней формы– TfrmMDIChild. Родительская форма содержит кнопку Clear в панели инструментов, которая удаляет содержимое memDailyNotes в активной дочерней форме. Вот как это реализуется.

```
procedure TfrmMDIParent.spbtnClearClick(Sender: TObject);
begin
  if not (ActiveMDIChild = Nil) then
     if ActiveMDIChild is TfrmMDIChild then
TfrmMDIChild(ActiveMDIChild).memDailyNotes.Clear;
end;
```

В первой строке проверяется, равен ли ActiveMDIChild значению Nil, так как в этом случае обращение к объекту вызовет исключительную ситуацию.

Поскольку ActiveMDIChild возвращает объект TForm, компилятор не имеет доступа к memDailyNotes – объекту TfrmMDIChild. Вторая строка проверят соответствие типов, т.е. действительно ли ActiveMDIChild указывает на объект TfrmMDIChild.

Третья строка выполняет преобразование типа и вызывает метод Clear компонента memDailyNotes.

MDIChildren и MDIChildCount

Свойство MDIChildren является массивом объектов TForm, предоставляющих доступ к созданным дочерним формам. MDIChildCount возвращает количество элементов в массиве MDIChildren.

Обычно это свойство используется при выполнении какого-либо действия над всеми открытыми дочерними формами. Вот код сворачивания всех дочерних форм командой Minimize All.

```
procedure TFormI.mnuMinimizeAllClick(Sender: TObject);
var
  iCount: Integers;
begin
 for iCount:= MDIChildCount-1 downto 0 do
 MDIChildren[iCount].WindowState:= wsMinimized;
end;
```

Если вы будете сворачивать окна в порядке возрастания элементов массива, цикл будет работать некорректно, так как после сворачивания каждого окна массив MDIChildren обновляется и пересортировывается, и вы можете пропустить некоторые элементы.

TileMode

Это – свойство перечислимого типа, определяющее, как родительская форма размещает дочерние при вызове метода Tile. Используются значения tbHorizontal (по умолчанию) и tbVertical для размещения форм по горизонтали и вертикали.

WindowMenu

Профессиональные MDI-приложения позволяют активизировать необходимое дочернее окно, выбрав его из списка в меню. Свойство WindowMenu определяет объект TMenuItem, который Delphi будет использовать для вывода списка доступных дочерних форм.

Для вывода списка TMenuItem должно быть меню верхнего уровня. Это меню имеет свойство Caption, равное Window.

MDI-события TForm

В MDI-приложении событие OnActivate запускается только при переключении между дочерними формами. Если фокус ввода передается из не MDI-формы в MDI-форму, генерируется событие OnActivate родительской формы, хотя ее свойство Active никогда и не устанавливается равным True. Эта странность на самом деле строго логична: ведь, если бы OnActivate генерировался только для дочерних форм, не было бы никакой возможности узнать о переходе фокуса ввода от другого приложения.

MDI-методы TForm

Специфичные для MDI-форм методы перечислены ниже. Arrangelcons выстраивает пиктограммы минимизированных дочерних форм в нижней части родительской формы.

Cascade располагает дочерние формы каскадом, так что видны все их заголовки. Next и Previous переходит от одной дочерней формы к другой, как будто вы нажали <Ctrl+Tab> или <Ctrl+Shift+Tab>. Tile выстраивает дочерние формы так, что они не перекрываются.

Работа с меню

Существуют определенные особенности работы меню в MDI-приложениях. Покажем, каким образом, MDI-приложение позволяет своим дочерним формам совместно использовать одну и ту же строку меню с помощью технологии *слияния меню*.

Если, к примеру, в строке меню дочерней формы TmdiEditFrm находятся два отдельных меню – File, Edit, Character. Строка меню родительской формы TmainForm содержит два элемента –File и Window. При активизации дочернего окна видно меню из четырех элементов – File, Edit, Character и Window. Меню File родительской формы и меню File дочерней формы содержат различные команды. Для того, чтобы получить этот эффект, следует использовать свойство GroupIndex. В нашем случае свойство GroupIndex меню File должно иметь значение 0, а это же свойство меню Edit, Caracter – значения1. Как и в случае дочерней формы, свойство GroupIndex меню File родительской формы имеет значение 0. Однако значение свойства GroupIndex меню Window – 9.

Свойство GroupIndex играет большую роль, так как именно оно управляет процессом слияния меню. Это означает, что, когда главная форма запускает дочернюю форму, меню последней сливается с меню главной формы. Свойство GroupIndex определяет, в каком порядке следуют отдельные меню и какие меню главной формы заменяются меню дочерней формы. Слияние применяется лишь к элементам строки меню компонента TmainMenu, а не к их командам.

Если значение свойства GroupIndex элемента меню дочерней формы совпадает со значением свойства GroupIndex элемента меню главной формы, элемент меню дочерней формы заменяет собой элемент меню главной формы. Оставшиеся меню располагаются в строке в порядке, определенном значениями свойств GroupIndex элементов объединенного меню.

Слияние меню в MDI-приложениях выполняется автоматически. Если значения свойств GroupIndex элементов меню установлены в

требуемом порядке, всегда будет присходить корректное слияние элементов мню при вызове дочерних MDI-форм.

Добавление в меню списка открытых документов

Для добавления в меню Window списка открытых документов необходимо присвоить свойству WindowMenu главной формы экземпляр элемента меню, содержащего требуемый список открытых документов. Например, свойству TmainForm.WindowMenu присваивается объект mmiWindow, ссылающийся на меню Window в строке меню приложения.

Для того, чтобы обеспечить отображение панели инструментов в родительском окне, нужно написать для дочерних форм следующие обработчики.

```
procedure TMDIChildForm.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  Action := caFree;
  {Переопределение объекта parent панели инструментов }
  tlbMain.Parent := self;
  { Если это была последняя дочерняя форма, то делаем видимой
панель инструментов главной формы }
  if (MainForm.MDIChildCount = 1) then
    MainForm.tlbMain.Visible := True
end;
```

В обработчике события TMDIChildForn.FormClose() параметру Action назначается значение caFree с целью гарантированного уничтожения экземпляра формы TMDIChildForm при ее закрытии. Переменная Action перечислимого типа может получить одно из четырех допустимых значений:

- caNone. Ничего не выполняется.
- caHide. Форма удаляется с экрана, но не уничтожается.
- caFree. Форма освобождается.
- caMinimize. Форма минимизируется (выполняется по учолчанию).

```
procedure TMDIChildForm.FormActivate(Sender: TObject);
begin
  { Когда форма становится активной, панель инструментов главной
формы следует убрать, после чего назначить и вывести в родительской
форме панель инструментов данного дочернего окна }
```

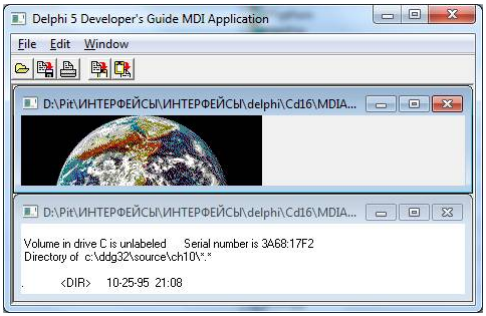
```
  MainForm.tlbMain.Visible := False;
  tlbMain.Parent := MainForm;
  tlbMain.Visible := True;
end;

procedure TMDIChildForm.FormDeactivate(Sender: TObject);
begin
  { Дочерняя форма становится неактивной либо при ее уничтожении,
либо в том случае, когда активной становится другая дочерняя форма.
Убираем панель инструментов этой формы, чтобы можно было сделать
видимой панель инструментов другой формы }
  tlbMain.Visible := False;
end;
```

Когда дочерняя форма становится активной, вызывается ее обработчик события OnActivate(). Всякий раз, когда дочерняя форма становится активной, выполняются определенные действия. Прежде всего, делается невидимой панель инструментов главной формы, что позволяет сделать видимой панель инструментов дочерней формы. Кроме того, главная форма назначается родительским объектом панели инструментов дочернего окна – поэтому данная панель инструментов будет отображаться в окне главной формы, а не дочерней. В обработчике события OnDeactivate() панель инструментов дочерней формы делается просто невидимой. Наконец, при обработке события OnClose() дочерняя форма вновь назначается в качестве родительского объекта ее панели инструментов, и если текущая дочерняя форма являлась единственной в приложении, то делается видимой панель инструментов главного окна. В результате при работе приложения создается впечатление, что главная форма содержит единственную панель инструментов, состав кнопок которой изменяется в соответствии с типом активного дочернего окна.

Задание

Создать интерфейс для программы «Просмотрщик файлов». Программа по желанию пользователя должна отображать как текстовые, так и графические файлы.



Вид приложения с двумя дочерними окнами.

3 Методические указания к выполнению самостоятельной работы

3.1 Общие положения

Целями самостоятельной работы является систематизация, расширение и закрепление теоретических знаний, приобретение навыков проектирования и реализации пользовательских интерфейсов с учетом человеческого фактора.

Самостоятельная работа по дисциплине «Проектирование человеко-машинного интерфейса» включает следующие виды активности студента:

- проработка лекционного материала;
- подготовка к лабораторным работам;
- изучение тем (вопросов) теоретической части дисциплины, вынесенных для самостоятельной подготовки;
- выполнение контрольных работ;
- подготовка к экзамену.

3.2 Проработка лекционного материала

Для проработки лекционного материала студентам рекомендуется воспользоваться конспектом лекции. Целесообразно ознакомиться с информацией, представленной в файлах, содержащих презентации лекций, предоставляемых преподавателем.

Рекомендуется сформулировать вопросы преподавателю и задать их либо посредством электронной образовательной среды вуза, либо перед началом следующей лекции.

3.3 Подготовка к лабораторным работам

Для подготовки к лабораторным работам студентам необходимо изучить соответствующий раздел настоящего пособия, выбрать среду программирования для создания интерфейса.

3.4 Изучение тем (вопросов) теоретической части дисциплины, вынесенных для самостоятельной подготовки

Перечень вопросов, подлежащих изучению

- Интерфейс Drag&Drop;

- Создание справочной службы системы.

3.4.1 Интерфейс DRAG&DROP

Операционная система Windows широко использует специальный прием связывания программ с данными, который называется Drag&Drop (перетащи и отпусти). Такой прием в Проводнике Windows используется для копирования или перемещения файлов. Этот интерфейс определяется двумя свойствами и тремя событиями, доступными каждому видимому компоненту.

Свойство

```
property DgagMode : TDragMode;
TGragMode = (dmManual, dmAutomatic);
```

Определяет, как будет выполняться весь комплекс действий, связанных с Drag&Drop: `dmManual` – вручную (программой); `dmAutomatic` – автоматически (свойствами и методами компонентов). Во всех случаях программист должен написать обработчики этих событий.

Свойство

```
Property DradCursor: TCursor;
```

Определяет вид указателя мыши в момент, когда над компонентом «протаскиваются данные». Если компонент готов принять данные, он устанавливает в это свойство значение `crDrag` , в противном случае – `crNoDrag`. Установка этих свойств осуществляется автоматически, если `DgagMode = dmAutomatic`.

Событие `OnDragOver(Sender, Soursce : TObject; X, Y: Integer; State: TDragState; var Accept: Boolean)` возникает в момент перемещения указателя мыши «с грузом» над компонентом.

Здесь `Sender` – компонент, который возбудил событие; `Soursce` – компонент-оправитель «груза»; `X, Y` – текущие координаты указателя мыши; `State` – состояние указателя (`dsDragEnter`– только что появился над компонентом; `dsDragLeave` – только что покинул компонент или была отпущена кнопка мыши; `dsDragMove` – перемещается над компонентом). В параметре `Accept` обработчик сообщает, готов ли компонент принять данные.

Событие `OnDragDropEvent(Sender, Soursce : TObject; X, Y: Integer)` означает, что пользователь «бросил» данные на компонент. Параметры обработчика совпадают по назначению с одноименными параметрами `OnDragOver`.

Наконец, при завершении перетаскивания (вне зависимости от того, приняты данные или нет) возникает событие

```
OnEndDrag(Sender, Target : TObject; X, Y: Integer);
```

где `Sender` – отправитель данных; `Target` – получатель данных или `NIL`, если никто не принял «посылку»; `X, Y` – координаты мыши в момент отпускания левой кнопки.

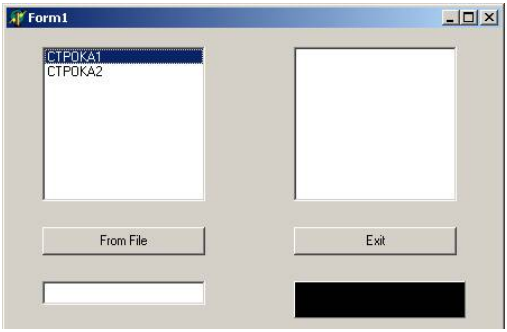
Задание 1.

Создать проект, содержащий компонент StringGrid и компонент Edit. Данный проект должен обеспечивать копирование строк из Edit в таблицу.

Задание 2.

Создать проект (см. пример интерфейса ниже), содержащий 2 списка ListBox, компонент Edit, компонент Panel и две кнопки. Данный проект должен обеспечивать следующие функции:

- копирование строк из Edit1 в первый список;
- перенос строк из первого списка во второй;
- удаление строк из первого списка методом их буксировки на черную панель.



3.4.2 Создание справочной службы системы

Для создания справочной системы рекомендуется воспользоваться одним из нижеперечисленных средств, доступных в открытых источниках сети Интернет: DrExplain, HelpAndManual, HelpMaker7, HelpNDoc и т.п.

3.5 Выполнение контрольной работы

«Создание интерфейса для решения задачи «Геометрия на плоскости»

Цель работы

Приобретение практических навыков для построения интерфейсов непосредственного манипулирования

Задание

При реализации интерфейса следует предусмотреть:

- Специальное окно, содержащее компонент PaintBox для графического представления данных.
- Таблицу StringGrid для табличного представления данных.
- Главное меню со следующими пунктами:
  - Данные → Создать – ввод нового набора данных;
  - Данные → открыть – чтение набора исходных данных из текстового файла (использовать OpenDialog);
  - Расчет – вычисления по условию задания;
  - Результат → Рисунок – представление результата на графике;
  - Результат → Файл – вывод результата расчета в текстовый файл (использовать SaveDialog);
  - Справка → О программе – окно с текстом условия задания.
- Кнопку, запускающую процедуру расчета.
- Рекомендуемый размер графической области 400*400 пикселей для квадрата ($-100 \le x \le 100, \quad -100 \le y \le 100$).
- Возможность добавления новых точек щелчком по графику.
- Возможность изменением в таблице.
- Механизм переноса точек по графической области.

Варианты заданий:

1. Дано N точек на плоскости. Выбрать из них три такие, чтобы была наибольшая площадь треугольника с вершинами в этих трех точках.
2. Дано N точек на плоскости. Выбрать из них три такие, чтобы был наибольшим периметр треугольника с вершинами в этих трех точках.
3. Дано N точек на плоскости. Найти окружность, на которой лежат наибольшее число точек данного множества.

4. Дано N точек на плоскости. Выбрать из них три такие, чтобы был наименьшим периметр треугольника с вершинами в этих трех точках.

5. Дано N точек на плоскости. Выбрать из них три такие, чтобы была наименьшая площадь треугольника с вершинами в этих трех точках.

6. Дано N точек на плоскости. Выбрать из них четыре такие, чтобы был наибольшим периметр прямоугольника с вершинами в этих трех точках.

7. Дано N точек на плоскости. Выбрать из них четыре такие, чтобы был наименьшим периметр прямоугольника с вершинами в этих трех точках.

8. Дано N точек на плоскости. Выбрать из них три такие, чтобы число точек, лежащих внутри треугольника с вершинами в этих трех точках, было максимальным.

9. Дано N точек на плоскости. Выбрать из них три такие, чтобы различались наименьшим образом число точек, лежащих внутри и вне окружности, проходящей через эти три точки.

4 Методические указания к выполнению курсовой работы

4.1 ОБЩИЕ ТРЕБОВАНИЯ К КУРСОВОЙ РАБОТЕ

Темы заданий выдаются преподавателем из списка прилагаемых тем (см. Варианты заданий). Студент планирует свою работу над проектом самостоятельно, с учетом рекомендаций преподавателя. Студент обязан в рамках установленный занятий по курсовой работе докладывать преподавателю о проделанной работе и согласовывать промежуточные результаты проектирования.

Результаты курсовой работы студент оформляет в виде пояснительной записки, содержание которой должно соответствовать требованиям, изложенным в настоящем пособии.

4.2 СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ

В процессе выполнения курсовой работы студент должен выполнить следующие этапы работы:

4.2.1 Определение назначения системы и создание каталога пользователей

Необходимо определить назначение системы, ее функции, входные и выходные данные.

В каталоге пользователей следует описать группы предполагаемых пользователей. Для каждой группы нужно указать тип пользователей (случайный, регулярный, программист, оператор), уровень знаний в области информационных технологий, профессию (если это имеет значение) и др. информацию.

4.2.2 Создание каталога требований

Каталог требований включает в себя:

- требования (ограничения) к оборудованию, т.е. вычислительной технике;
- требования от технологии использования программного продукта (как часто будет использоваться, с какими системами должен сопрягаться, какой объем работ будет с его помощью выполняться и т.д.);
- требования от пользователей (стиль интерфейса на основе личных и профессиональных привычек, опыта и др.).

На основании анализа требований выбирается инструмент (программная среда) для создания Вашей системы, например, Lazarus, Delphi, Access, C++Builder.

4.2.3 Построение структуры диалогового взаимодействия

Выделите отдельные функциональные блоки, составляющие Вашу систему и отдельные блоки данных (файлы, базы данных).

Составьте спецификацию на каждый функциональный блок. Примерная структура спецификации приведена в следующей таблице.

Спецификация функционального блока

Атрибут	Значения
Назначение	Описание назначения блока, его цели
Описание	Общий обзор действий блока
Внешнее отображение	Связанные с блоком экранные элементы (окно)
Входные/выходные данные	Содержание входных и выходных потоков данных
База данных	Используемая блоком постоянная информация
Входные/выходные сигналы управления	Содержание входных и выходных сигналов управления и действий, осуществляемых по этим сигналам

Составьте структуру взаимодействия объектов (функциональных блоков, файлов и пользователей), в которой необходимо отразить потоки данных и потоки управления. Пример схемы взаимодействия приведен на рисунке:

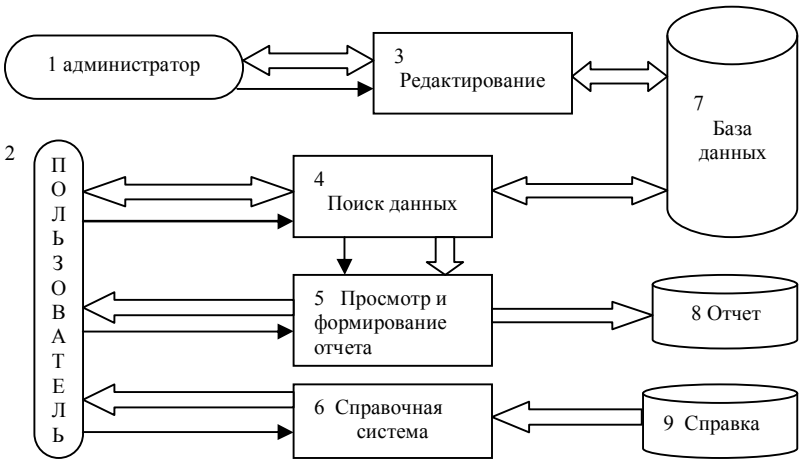


Схема взаимодействия объектов системы

Опишите все потоки данных и потоки управляющей информации. Обозначение потоков данных начинается с буквы D (Data), потоков управления - с буквы C (Control). После буквы через черточку указывается блок-отправитель и блок-получатель потока. Например, описание потоков данных и управления, которыми обмениваются блок 2 "пользователь" и блок 4 "поиск данных" может быть следующим:

C2-4 - команда проведения поиска

D4-2 - система запрашивает ввод шаблона поиска

D2-4 - пользователь вводит шаблон поиска

Можно привести функциональную модель системы, созданную по технологии IDEF0 или UML (диаграммы Use case и/или Activity) .

Вся информация о взаимодействии системы с пользователем может быть сведена к матрице "роль пользователя/ функция системы", структура которой приведена в таблице.

Матрица "роль пользователя/ функция системы"

Функциональный блок	Роль пользователя		
	управление	входные данные	выходные данные
Функция 1			
Функция 2			

4.2.4 Разработка интерфейсных объектов.

При разработке меню и экранных форм, в том числе окон, диалоговых панелей и др. Вы должны учитывать принципы создания дружественных интерфейсов, изложенные в лекционном материале:

- принцип минимального рабочего усилия;
- принцип экономии памяти пользователя;
- принцип минимального времени на обучение;
- принцип согласованности элементов интерфейса;
- принципы учета возможностей пользователя (уровня знаний в области ВС, профессиональных привычек, общепринятых традиций и ассоциаций, индивидуальных особенностей пользователя.);
- принцип отображения текущего состояния процесса
- принцип визуализации;
- принцип "помощи" пользователю
- принцип объяснения результатов
- принцип диагностики ошибок и отказов
- принцип контроля доступа
- принцип активности пользователя
- принцип открытости, изменяемости системы

Кроме того, Вы должны руководствоваться правилами использования цвета и форматирования экранных форм:

1. Данные должны располагаться так, чтобы пользователь мог просматривать их в логической последовательности. Как правило, направление просмотра - из левого верхнего угла слева направо и сверху вниз.

2. Данные должны располагаться так, чтобы пользователь мог идентифицировать связанные группы информации. Отдельные группы логически связанных данных можно отделять вертикальными и горизонтальными линиями, помещать в отдельные ниши, панели.

3. Информация должна располагаться так, чтобы окно было композиционно "уравновешенным", т.е. "центр тяжести" должен быть примерно посредине окна. Желательно также, чтобы информация не была слишком плотной, чтобы не утомлять пользователя.

4. Расположение одинаковой или сходной информации в различных окнах должно быть согласованно. Желательно использование единого шаблона. Во время проектирования изображений полезно нарисовать их на разлинованной бумаге. При этом те элементы, которые являются общими для различных изображений (например, кнопки Ok, Cancel) следует помещать в одно место.

5. Выбор цвета и цветовых сочетаний не должен быть хаотичным. Вот несколько советов по использованию цвета:

- используйте минимальное количество цветов (не более 3 - 4-х), т.к. слишком пестрые изображения быстро утомляют глаза;
- для фона лучше использовать более спокойные тона. Если в изображении используется большое количество цветов, фон лучше сделать белым или серым. На светлом фоне цвета кажутся ярче и легче воспринимаются при различном внешнем освещении;
- текст и изображение должны четко выделяться на фоне. Нельзя использовать желтый цвет на белом фоне и синий - на черном;
- некоторые комбинации неприятны для глаз, например, голубой цвет символов на красном фоне;
- нужно учитывать общепринятые представления о цветах. Например, красный цвет считается цветом опасности и его лучше использовать в сообщениях об ошибках.

При создании справочной службы Вашей системы используйте правила и советы по созданию системы справок:

1. Полезно организовывать систему справок таким образом, чтобы она имела древовидную структуру. Самый первый раздел должен содержать перечень основных разделов. Это своеобразный каталог (содержание) всей справочной информации. Любой раздел может

содержать список подчиненных разделов. Не следует делать слишком запутанную систему ссылок, это лишь затруднит пользователю поиск нужной справочной информации.

2. Каждый раздел, по возможности, должен полностью отображаться в распахнутом окне. Избегайте слишком длинных пояснений, длинный раздел лучше разбить на несколько связанных подразделов. Учтите, что люди считают текст на экране гораздо медленнее, чем напечатанный текст.

3. Старайтесь излагать справочную информацию простым и ясным языком с использованием примеров, иллюстраций. Можно использовать юмор, но не переусердствуйте в этом. Не используйте профессиональный жаргон.

4. Избегайте тесноты "монолитный" текст очень утомляет и затрудняет усвоение информации. Вставляйте пустые строки, абзацы, рисунки, выделяйте важную информацию шрифтом и цветом. Но помните, что слишком "раскрашенный" текст тоже может утомлять.

4.2.5 Программирование и отладка системы

В соответствии с разработанным проектом создайте систему с помощью выбранного Вами инструмента (программной среды).

4.3 Требования к оформлению отчетов

Рекомендуется следующее содержание отчета:

- титульный лист,
- содержание,
- введение,
- основная часть,
- заключение,
- список использованных источников;
- приложения.

Введение должно содержать цель курсовой работы, основные принципы, положенные в основу ее проведения, ее значение и область применения.

Основная часть работы должна отражать процесс и результаты проектирования пользовательского интерфейса. Примерное содержание основной части работы:

- назначение системы;
- основные функции системы;
- каталог пользователей;
- каталог требований к интерфейсу;
- выбор программного средства реализации;

- структура диалогового взаимодействия с пользователем;
- основные экранные элементы интерфейса;
- структура справочной системы.

Заключение должно содержать краткие выводы по результатам выполненной работы.

Список использованных источников оформляется согласно стандарту.

В приложении приводятся:
- руководство пользователя (обязательное приложение);
- распечатка программы или фрагмента, отражающего реализацию интерфейса (рекомендуемое приложение).

4.4 Варианты заданий

1. Электронный ежедневник
2. Справочная система «Библиография» для хранения и выдачи информации о книгах
3. Система резервирования авиабилетов
4. Справочная система «Документ» для хранения и печати документов различного вида
5. Справочная система «Успеваемость» для хранения информации о текущей успеваемости студентов
6. Телефонный справочник
7. Система в помощь переводчику с английского языка (перевод выделенных в тексте слов с помощью словаря, пополнение словаря)
8. Справочная система аэропорта
9. Диалоговая система по обмену жилья
10. Обучающая система (вывод обучающей информации по некоторой теме и проведение тестирования по данной теме)
11. Диалоговая система оценки знания правил дорожного движения
12. Справочная система по вузам для абитуриентов
13. Графическая система «Планировщик» для размещения мебели на плане комнаты (задание габаритов комнаты, мебели, передвижение с помощью «мыши» контуров мебели на плане)
14. Справочная система по курсам валют
15. Справочная система «Кулинария» для хранения и выдачи рецептов блюд (по выбранной пользователем категории, виду кухни)
16. Справочная система по туристическим маршрутам
17. Выбор места в самолете (отметка на плане салона самолета забронированного места и ввод информации о пассажире)
18. Система для определения соционического типа
19. Система «Склад» для хранения и выдачи информации о товарах

20. Психологический тест
21. Справочная система «Фильмотека» для хранения и выдачи информации о фильмах
22. Справочная система «Меломан» для хранения и выдачи информации об аудиоальбомах
23. Система «Бюджет семьи» для контроля доходов и расходов семьи
24. Генератор кроссвордов
25. Выбор места в театре (отметка на плане зарезервированного места и ввод информации о клиенте)

5 Список литературы

1. Акчурин Э.А. Человеко-машинное взаимодействие: Учебное пособие. –М.: СОЛОН-ПРЕСС, 2008.-93 с.

2. Логунова О.С., Ячиков И.М., Ильина Е.А. Человеко-машинное взаимодействие: Теория и практика. – Ростов-на-Дону: «Феникс», 2006.- 288 с.

3. Мандел Т. Разработка пользовательского интерфейса. – "ДМК Пресс", 2007. – 418с. URL: https://e.lanbook.com/book/1227#book_name

4. Клонингер К**.** Свежие стили Web-дизайна: как сделать из вашего сайта «конфетку». **–** "ДМК Пресс", 2009.- 250 с.
URL: https://e.lanbook.com/book/1067#book_name

5. Ткаченко О.Н. Взаимодействие пользователей с интерфейсами информационных систем для мобильных устройств: исследование опыта: уч. пособие.– М.: Магистр: ИНФРА-М, 2018.– 152 с.
URL: http://znanium.com/bookread2.php?book=937425